What's New in TeamCity 2017.2

100 Build Configurations in TeamCity Professional

Starting from this version, the free TeamCity Professional edition allows 100 build configurations per server.

TeamCity-Docker Integration

The built-in TeamCity-Docker integration is now available.

Supported Platforms

TeamCity-Docker support can run on Mac, Linux, and Windows build agents.

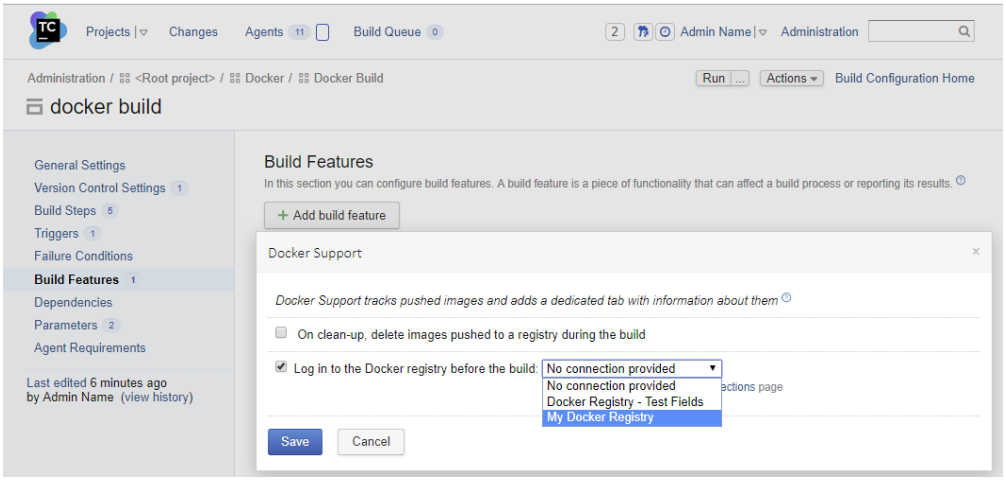
- On Linux, the integration will run if the installed Docker is detected.
- On Windows, the integration works in the Windows container mode only. Docker on Windows with the Linux container mode enabled is not supported, an error is reported in this case.
- On MasOS, the official Docker support for Mac should be installed for the user running the build agent.

The integration provides several features to facilitate working with Docker under TeamCity.

Docker Runners

- Docker Wrapper is an extension for Command Line, Maven, Ant, and Gradle runners enabling you to run the build step in an isolated Docker container. Each of the supported runners has a dedicated Docker settings section.
- Docker Build Runner allows building Docker images as a separate build step. When creating TeamCity projects/ build configurations from a repository URL, the runner is offered as build step during auto-detection, provided a Dockerfile is present in the VCS repository.
- Docker Compose Runner starts docker-compose build services and shuts down those services at the end of the build.

Docker Support Build Feature



This build feature serves the following purposes:

- Monitoring Docker events and providing information about Docker-related operations: the Docker Support build feature adds the dedicated Docker Info tab to the build results page.
- Authentication to a Docker registry can be configured at the project level to be easily shared among builds. With this configured, you can use Docker Support to instruct the agent to automatically log in to a registry before the build and log out of it after the build.
- It is also possible to enable automatic clean-up of images published by a build during the server clean-up.

Another clean-up-related feature of the integration is the ability of Free Disk Space to run the `docker system prune -a` command cleaning the local Docker Caches if there is not enough disk space on the agent at the beginning of the build.

.NET CLI Support

TeamCity 2017.2 can now build, test, and deploy .NET Core projects out of the box: one of the highly popular TeamCity plugins, .NET CLI Support (aka .NET Core), is now bundled with TeamCity. The plugin has been significantly reworked and improved in terms of functionality and usability. It simulates the behavior of the dotnet command, which is fully supported out of the box now, and provides the following features:

- dedicated .NET CLI build runner
- automatic discovery of build steps when given the URL of your repository
- detection of .NET CLI on the build agents
- hierarchical build log
- real-time reporting of tests, compilation errors or other problems
- code coverage powered by JetBrains dotCover

Composite Builds

Prior to TeamCity 2017.2, if you used build chains, there was no easy way to see whether the chain started or not. And if it did, you could not see failed tests and other build problems of the whole chain in one place. In addition, users often created a dedicated build configuration to aggregate artifacts of several builds in a chain and present them in a single place.

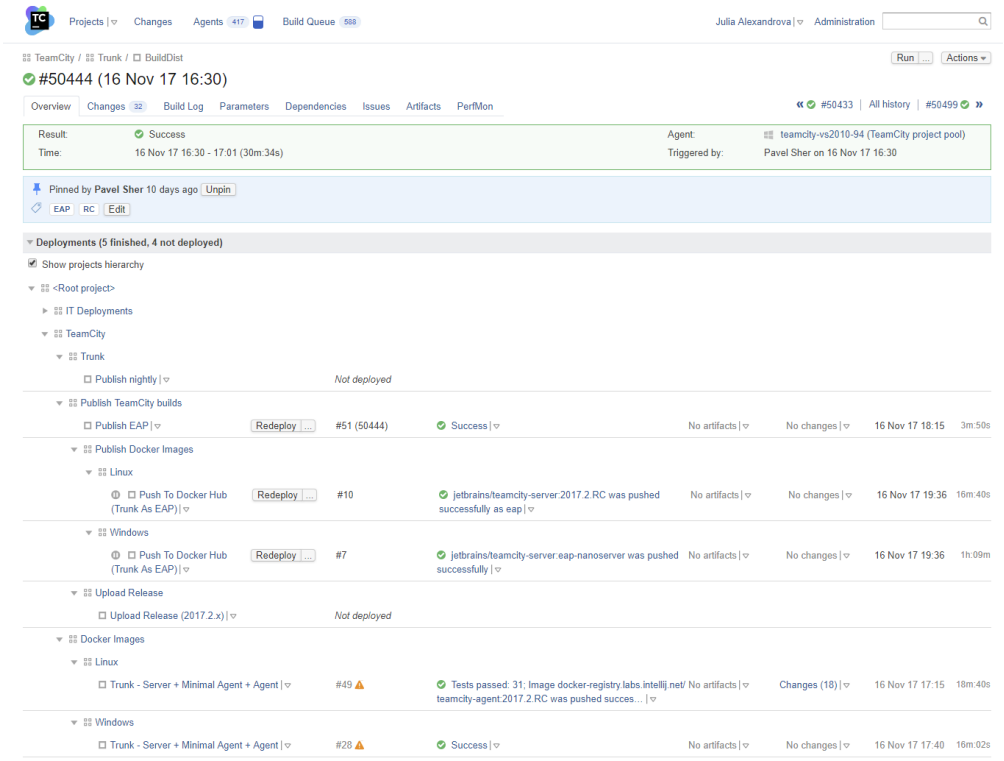
To ease this pain for the users, TeamCity now has the composite build configuration. Its purpose is to aggregate results from several other builds combined by snapshot dependencies and present them in a single place.

A composite build does not occupy an agent; it is shown as running at the same time when the first dependency of the build chain starts, and is shown as finished when the last dependency finishes. More details are available in our documentation.

Deployment Build Configuration

TeamCity users often create build configurations performing deployment tasks. Such configurations usually have snapshot or artifact dependencies on the builds, whose results they deploy.

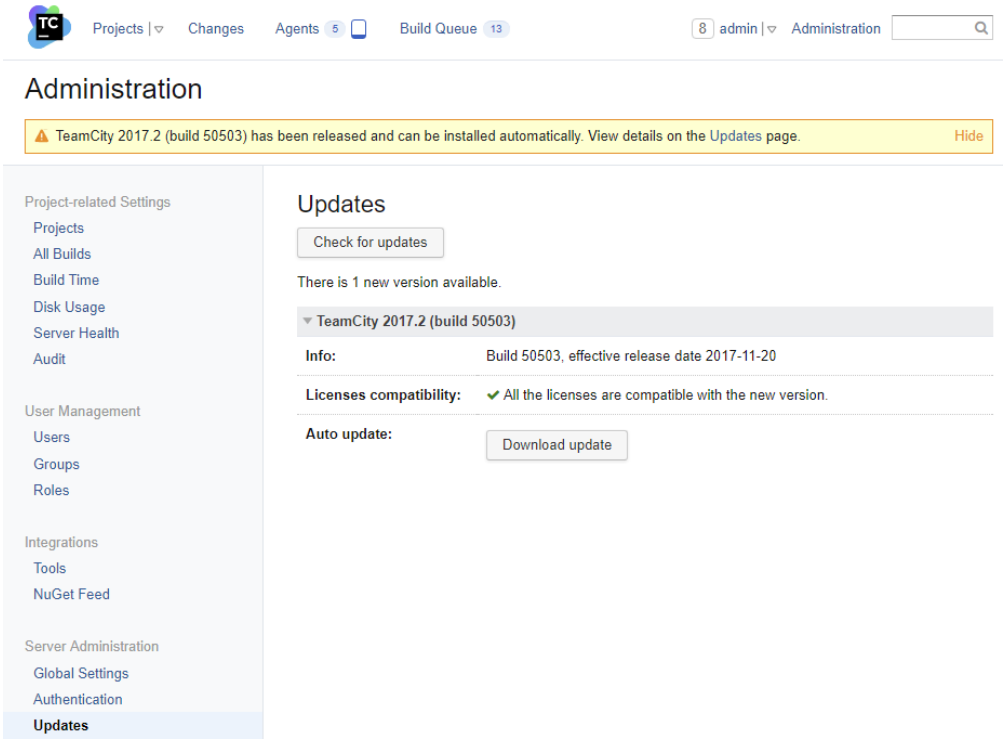
TeamCity 2017.2 allows you to mark such configurations as Deployment. After that, the builds they depend on, get the dedicated Deployments section on the build results page of this build, allowing you to quickly deploy the build:



More details are available in our documentation.

Server Auto Update

Upgrading your TeamCity server is much simpler now. When a new version of TeamCity is detected, the server displays the corresponding health item for system administrators pointing to the new Updates page in server Administration. The page lists all the related options as well as notes about licenses compatibility, the new version description and controls to perform automatic update. All you need to do is follow the instructions on the page. Details and limitations are described in our documentation.



Managing Plugins from Web UI

The new options improve plugins management in TeamCity. There are now dedicated options on the Administration | Plugins List page to delete or disable plugins. Besides, if you upload a plugin, TeamCity will offer you to restart the server to apply your changes.
Now there is also the server Restart button on the Administration | Diagnostics page.

Default Build Configuration Templates for Project

You can now use an existing template and set it as the default template of a project. After that, any template modification will automatically affect all build configurations in this project and its subprojects. Now it is much easier to add a specific build feature to all build configurations of a project, or switch all build configurations to some specific checkout mode, or provide a default failure condition.

Multiple Templates for Build Configuration

Now a build configuration can be attached to multiple templates using the "Attach to template..." action menu. There is a new menu item in the Actions menu, Manage templates, allowing users to manipulate templates and build configurations. See details in our documentation.

Kotlin DSL Improvements

This TeamCity version provides several features, requested by the users, most wanted being the ability to edit settings via the TeamCity Web UI and Kotlin DSL documentation.

Editable Administration UI for Kotlin DSL projects

Earlier, when Kotlin DSL was enabled for a project, the administration part of the TeamCity user interface used to be read-only which was challenging if you just started working with Kotlin projects.

Now when you switch a project to the Kotlin DSL, project and build configuration settings will be available for editing from the web UI. All changes made in a project via the UI will be presented as a patch in the Kotlin format which will be checked in under the project "patches" directory. Details and limitations are described in our documentation.

If you are already using Kotlin DSL, to use editing from the UI, you need to migrate to the new DSL API version.

DSL API Documentation

Now after Kotlin DSL is enabled for a project, the TeamCity server provides html documentation for the core DSL API as well as the API provided by installed TeamCity plugins. The documentation is accessible via the link on the 'Versioned Settings' project tab in the UI or by running the `mvn -U dependency:sources` command in the IDE.

Other Changes

Other benefits of migrating to the newer version of DSL API include:

- better completion in the IDE: Kotlin 1.1 introduced a way to limit functions available in a particular scope, which allows having a better completion in the IDE.
- the ability to validate settings before applying them to the TeamCity server. Plugins now mark their mandatory properties and TeamCity will reject settings from the VCS without such properties specified (e.g. a Git VCS root without the url). You can also provide a custom validation logic by overriding the `validate()` method in your classes.
- the ability to use external libraries in your Kotlin DSL code, which allows sharing code between different Kotlin DSL-based projects
- the dedicated DSL for new plugins bundled with TeamCity
- debugging of the Maven task generating TeamCity configs.

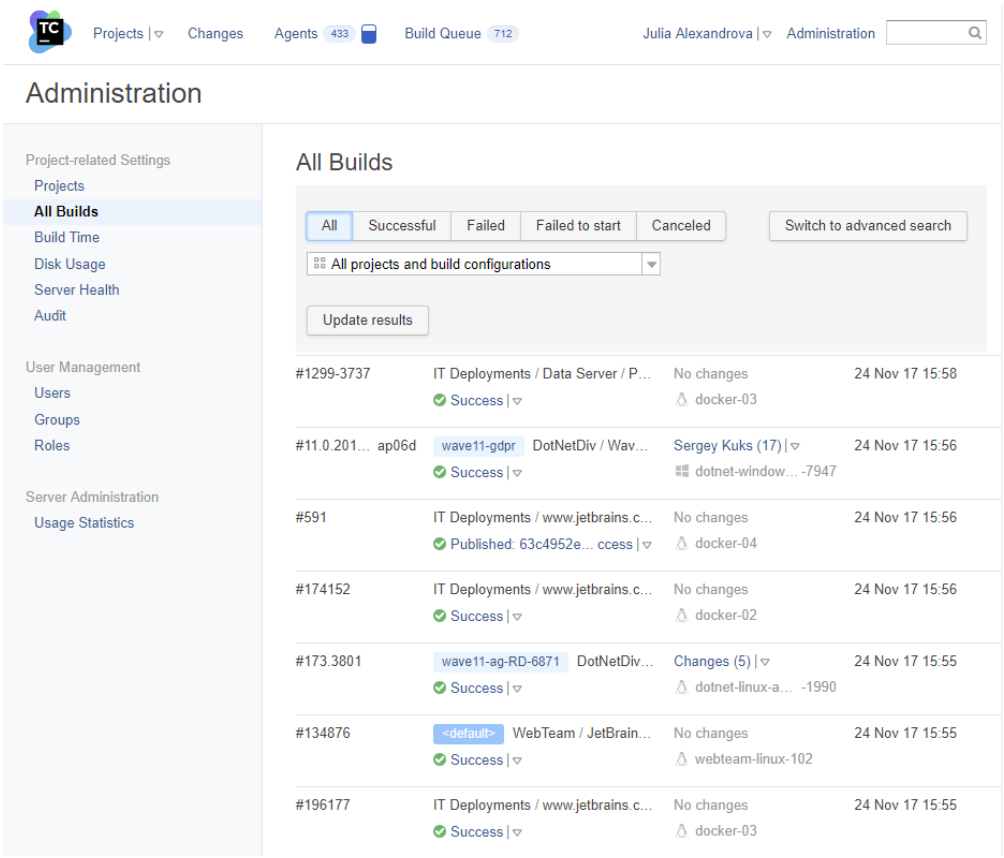
UI Improvements

TeamCity 2017.2 comes with polished web UI, bringing you a number of improvements. We also unveil a set of pages which use components built with React and serve data provided by the TeamCity REST API.

All Builds page

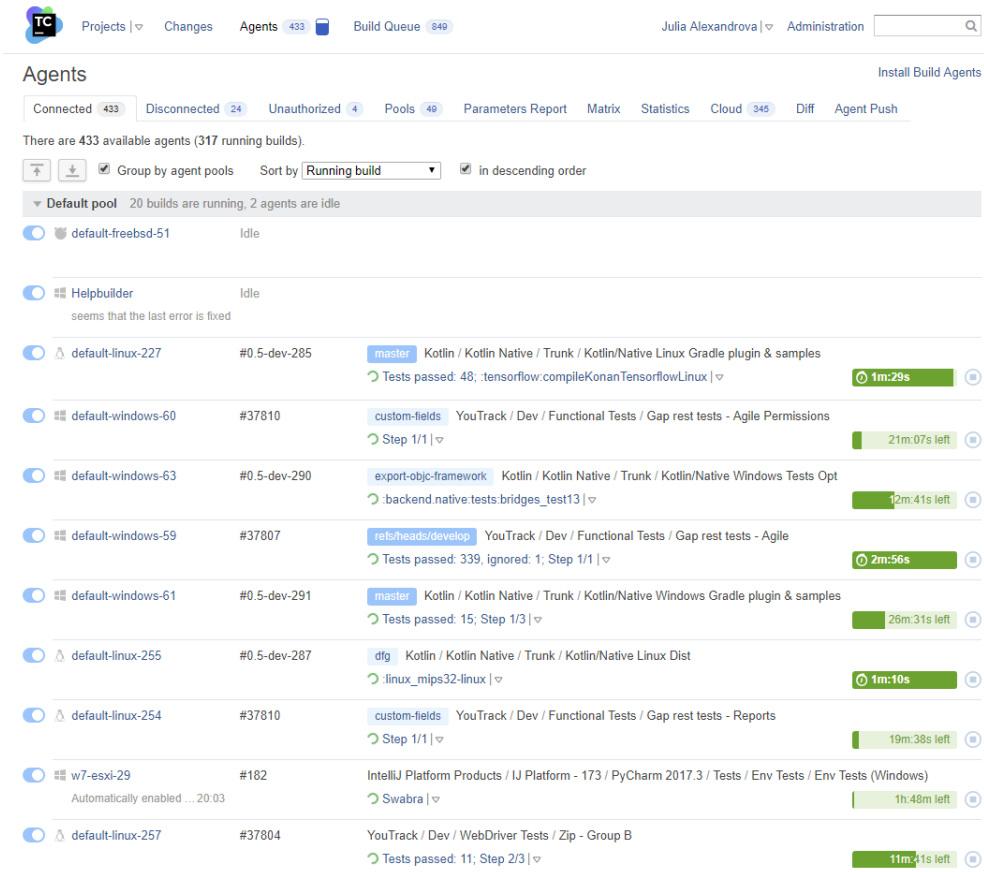
The All Builds page allows server administrators view and filter finished builds. By default, the builds can be filtered by build status: Successful, Failed, Canceled and Failed to start, as well as by project or build configuration. Since the results are obtained via the TeamCity REST API, all build locators are supported by the advanced search options.

This page is especially useful right after the upgrade of the server. Using this page, system administrators can find recently failed builds faster, which in turn helps to identify compatibility problems which may occur right after the server upgrade.



New Look for Agents page

The Agents page is also built using the data provided by the TeamCity REST API and boasts a new, fresh look. The page is optimized to work fine with hundreds of agents and not to load the browser CPU.



REST API Improvements

TeamCity REST API new capabilities and improvements include:

- Managing build configuration's multiple templates is fully supported
- Managing project's default template is supported
- Assigning mutes and investigations for tests and build problems
- Ability to create a user group with all the details (child groups and roles) via a single request
- Improved performance for some requests, including filtering builds by `"agent"`, by `"affectedProject"` or multiple values of the `"buildType"` dimension.

Bundled Tools Updates

- Tomcat has been updated to version 8.5.23
- The bundled .net Tools, dotCover, and ReSharper Command Line Tools have been updated to the latest released version, 2017.2.2
- The bundled JRE updated to 1.8u151
- TFS Java SDK has been updated to 14.119.2.

⚠ Be informed that starting from v14.114.0, Microsoft removes support for the following environments:

- Java 1.6
- AIX
- HP-UX
- Windows Vista
- Eclipse versions prior to 4.2
- Team Foundation Server 2010

Other Improvements

- Commit status publisher now supports Git repositories hosted in TFS/VSTS and supports publishing commit as well as pull request statuses.
- TeamCity now automatically recognizes commits to GitHub, VSTS or Bitbucket and provides links enabling you to view the changes externally, without configuring an external changes viewer per a VCS Root
- Now you can centrally manage NUnit console versions installed on agents using the Administration | Tools page. Different versions of NUnit 3 can be installed and will be automatically distributed to all build agents.
- Now the TeamCity server will restart automatically if the Java process of the server crashes.
- Perforce Stream Depth Support: starting from this release, TeamCity supports deeper directory structure within the root depot: depots with a depth of //DEPOTNAME/1/2/n can be specified in the Perforce Stream field.
  Experimental support for native SVN checkout has been added
- Cloud Agent Support enhancements include:
  - Now you can configure a custom URL that the agents cloned from the image will use to connect to the TeamCity server. This URL has to be available from the build agent machine. By default, the TeamCity server URL specified on the Global Settings page of the Administration UI is used.
  - Manual termination: when using cloud agents, you can now choose to stop the instance after the current build using the corresponding new option on the agent page, and the agent will terminate gracefully. It is also possible to force instance termination if required.

Fixed Issues

- Full list of fixed issues

Previous Releases

What's New in TeamCity 2017.1